Project Work & Exam Rules

Hospital Management

Java Advanced Programming

a.y. 2024-2025

Rev. 1.0

Project Work Statement

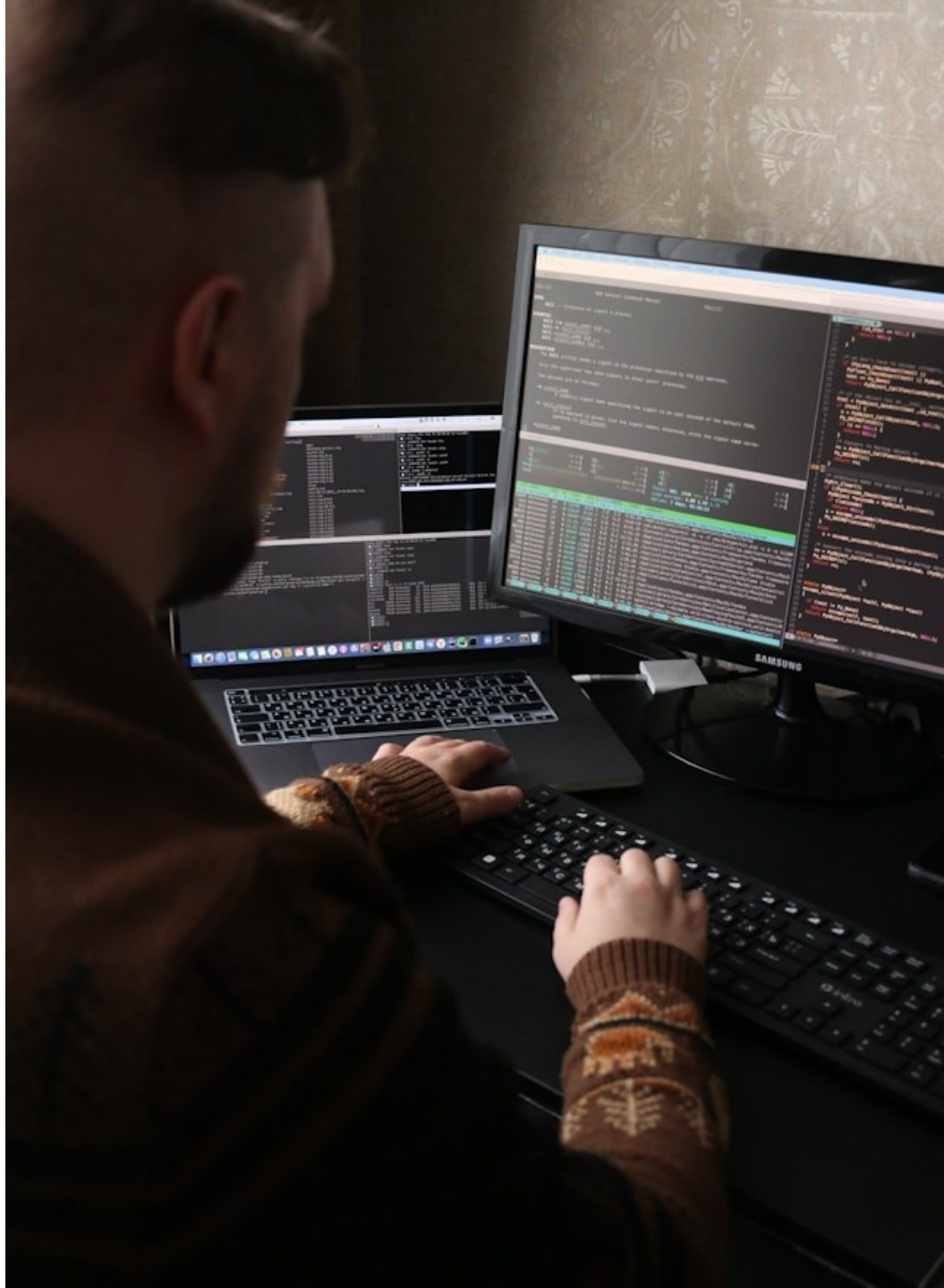
- Develop a software to record the admissions and discharges of patients at “Polis Healthy Hospital” (PHH).
- The project should be made in Java Spring Boot
- Database should be a relational one like MySQL or PostgreSQL
- In the following slides there are the “complete” UML specifications of the project
- Unit tests using Junit must be provided for at least five services (SpringBoot service layer)

Project Work Statement/2

- Hospital is divided into departments
- A patient can be, at a given time, in one department only
- Admission, discharge and department change of a patient must be traced
- The discharge can happen for one of three possible causes: patient is healthy again, patient needs to be transferred to another hospital, patient is dead
- Clinical data is a free text where medical doctors and nurses write some text

Project Work Statement/3

- An entity with relationships one-to-many (it is the one) cannot be deleted if there is at least one associated record on the many side. An error message must be displayed. E.g. if a department has at least one patient associated, it cannot be deleted.
- The errors management has not been considered in this specifications and must be defined by the students.
- Errors that appear on client side must be clear and user-friendly. On server side al exceptions must be caught and managed in some way.
- Errors displayed on client-side must be defined at server-side level and propagated towards the client



Groups

- Each project work can be done by a group composed from 1 to 4 students
- All the members of the group must be able to explain in detail both the server-side code
- The group will present its project work once, demonstrating its functionalities and how the automated testing scripts detect problems (at least one example)
- Each group member (i.e. each student) will be individually interviewed to check if he or she knows the code in detail and if he or she is able to explain randomly chosen lines of code

Code rules

- Server code must be developed in Java + SpringBoot
- Server code must be put in a private GitHub repository
- Every repository must have as collaborator the GitHub user *luca-lezzerini (to let the teacher to see the code)*
- Cloning the repositories must produce runnable code (only libraries dependencies download can be tolerated)
- Database should be MySQL or PostgreSQL

Rules for JAP exam

- 10 points for attendance
- 10 points for group project work evaluation (threshold 5)
- 40 points for written test (threshold 20)
- 40 points for individual interview about the code (threshold 20)
- Code must be delivered on the day before the exam sending the links of all the repositories (client and server) by mail to luca_lezzerini@universitetipolis.edu.al
- Group presentation must be sent the day before the exam to the same e-mail address above

Common rules

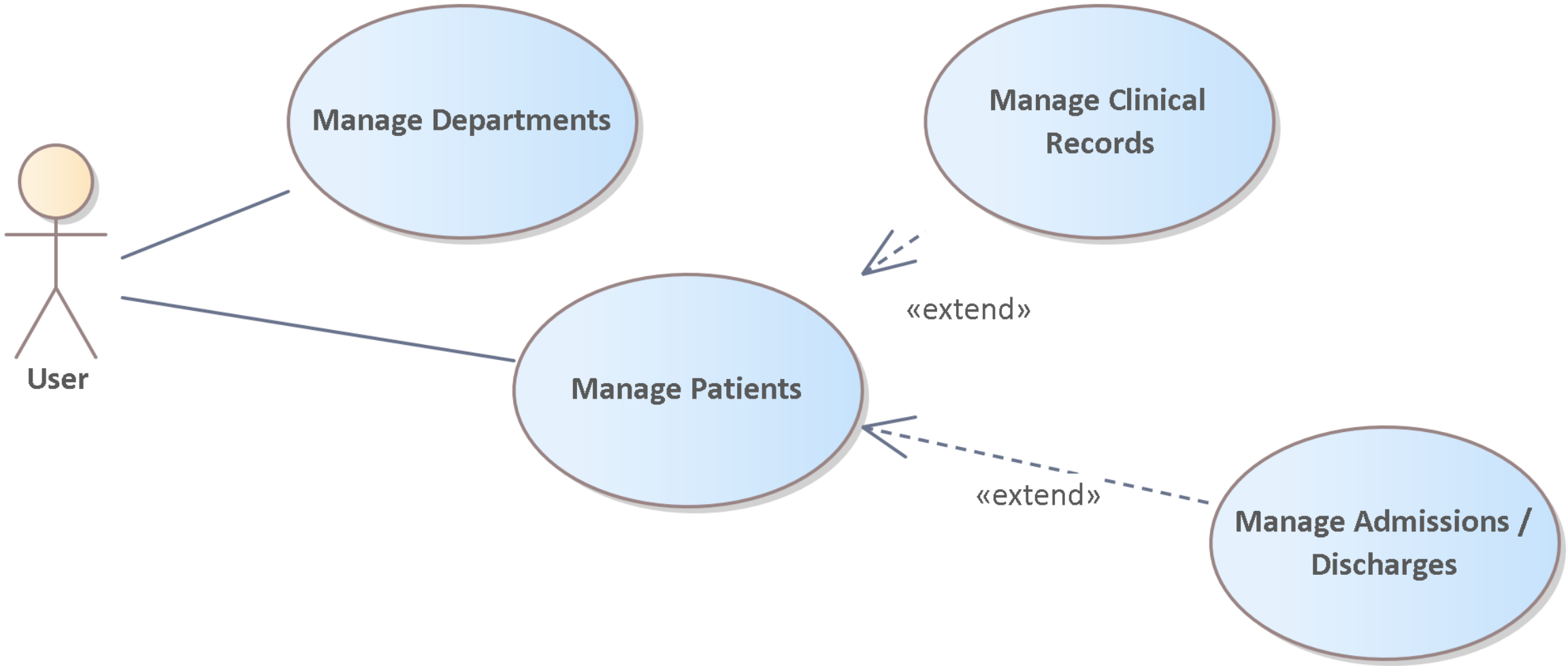
- Groups can attend the exam with their own computer with the completely configured project work
- If a group cannot use their own computer, then they must come with an external hard disk with a virtual machine configured to run the project work (VirtualBox or VMWare)
- Students that will fail the exam can repeat it in another session
- **The minimum threshold must be reached for each part of the exam to pass the exam**. If one or more elements of the exam have been evaluated below the threshold, the student will have failed the exam.

Project Work Presentation

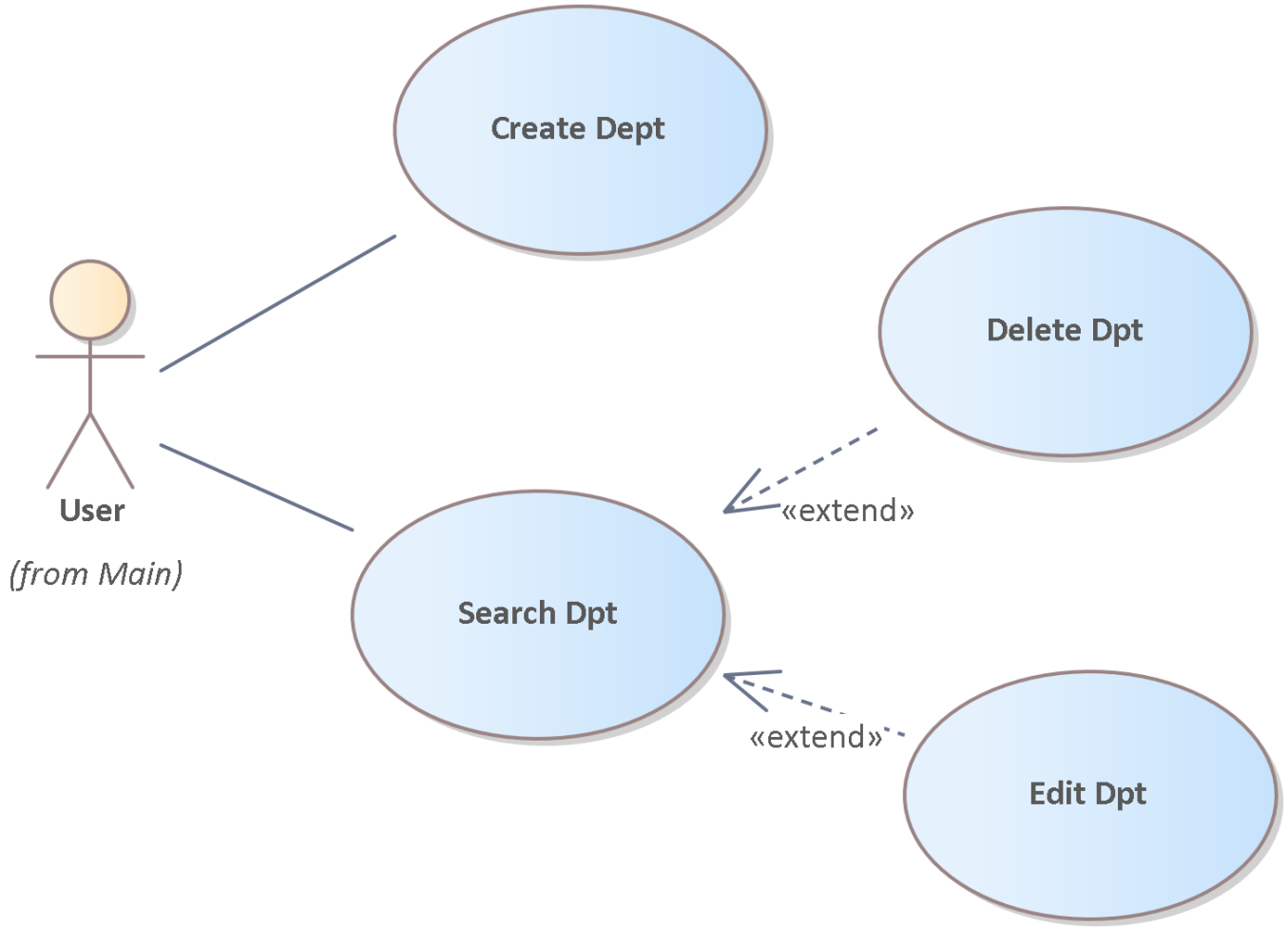
- The project work must be presented using a presentation (e.g. PowerPoint) where are reported, at least, the following information:
  - List of all RESTful services
  - List and description of all automated tests for unit testing
  - Sample screenshots of the application
  - Screenshots of the execution of the automated unit tests
  - UML diagrams for controllers, services, DTOs and entities

Use Cases

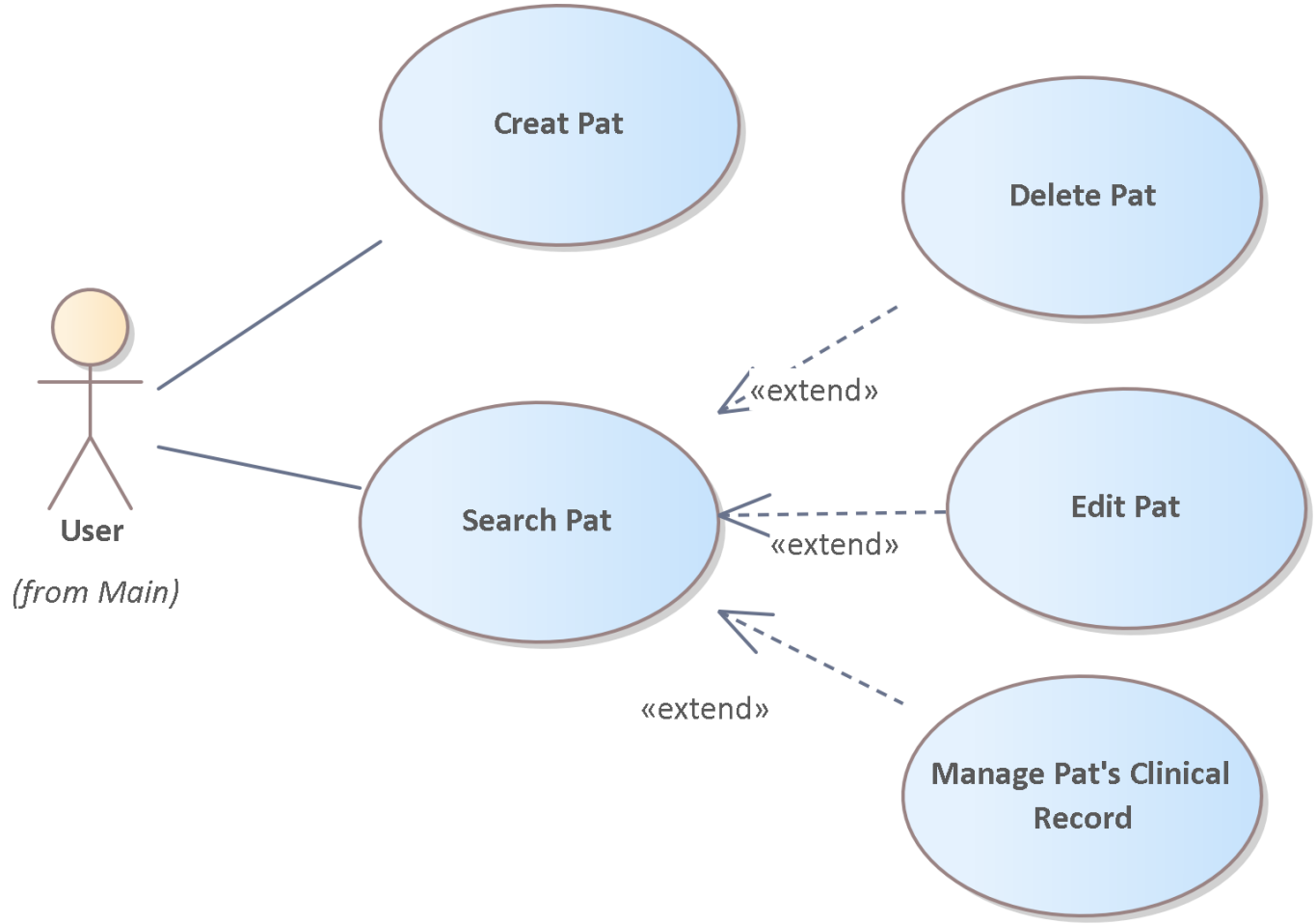
Main UC


Manage Departments
Manage Clinical Records
User
«extend»
Manage Patients
«extend»
Manage Admissions / Discharges

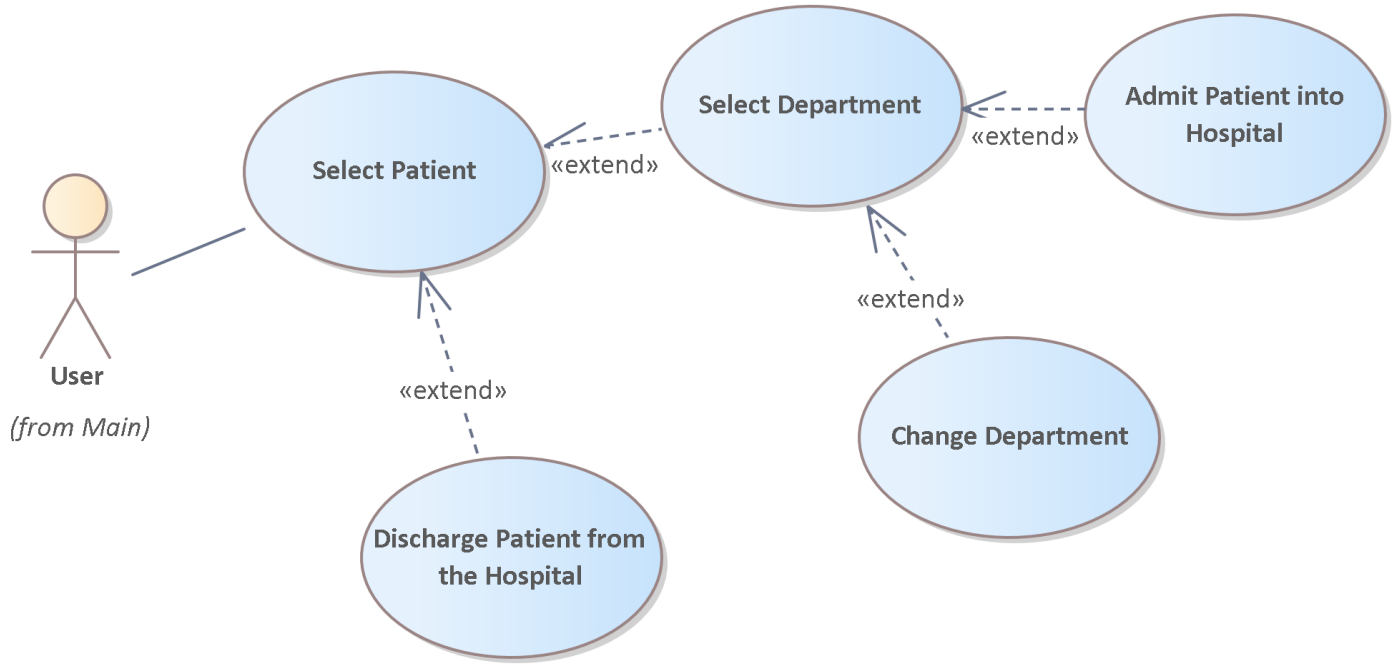
Manage Departments



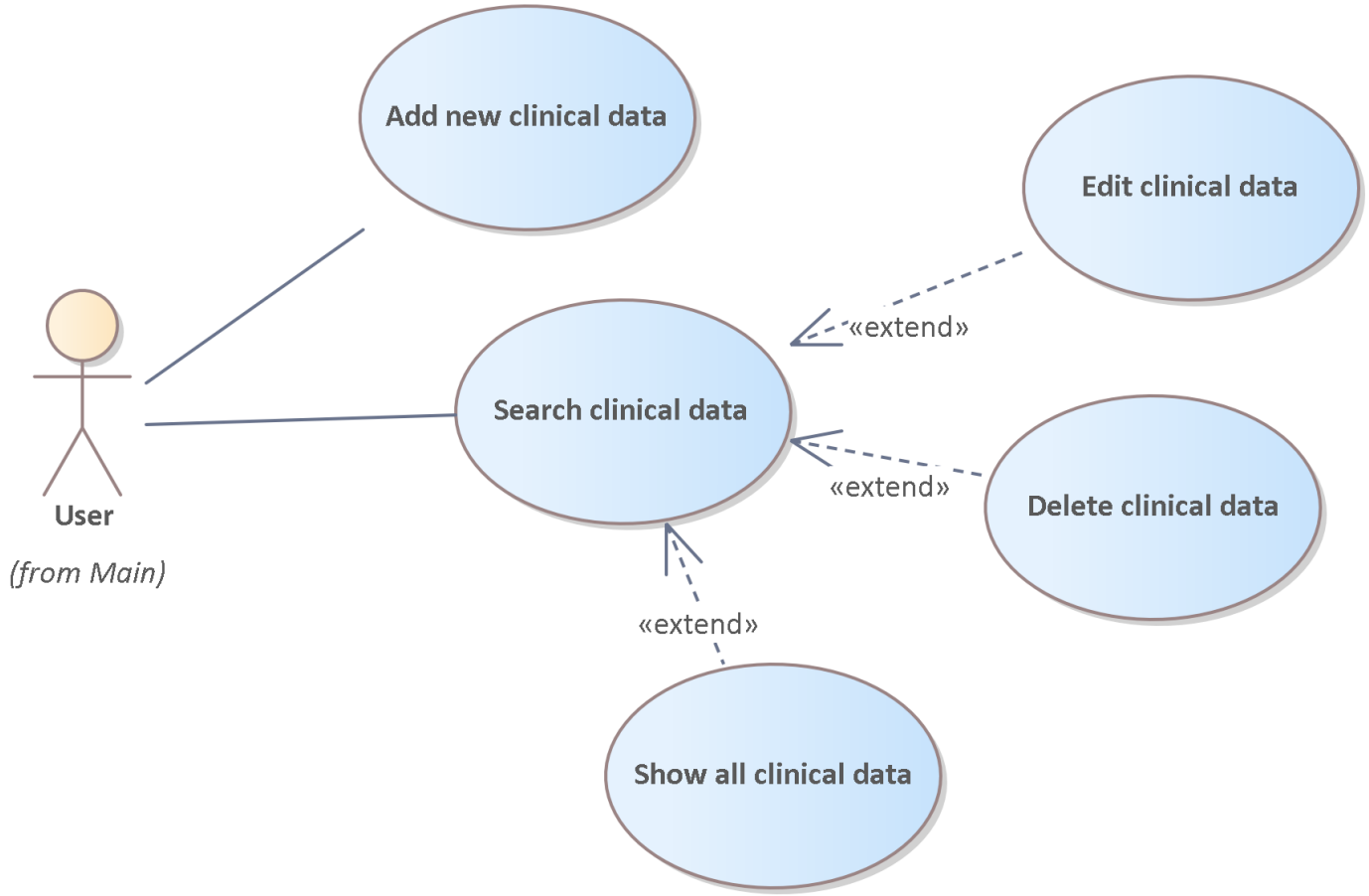
Manage Patients



Manage Admissions / Discharges

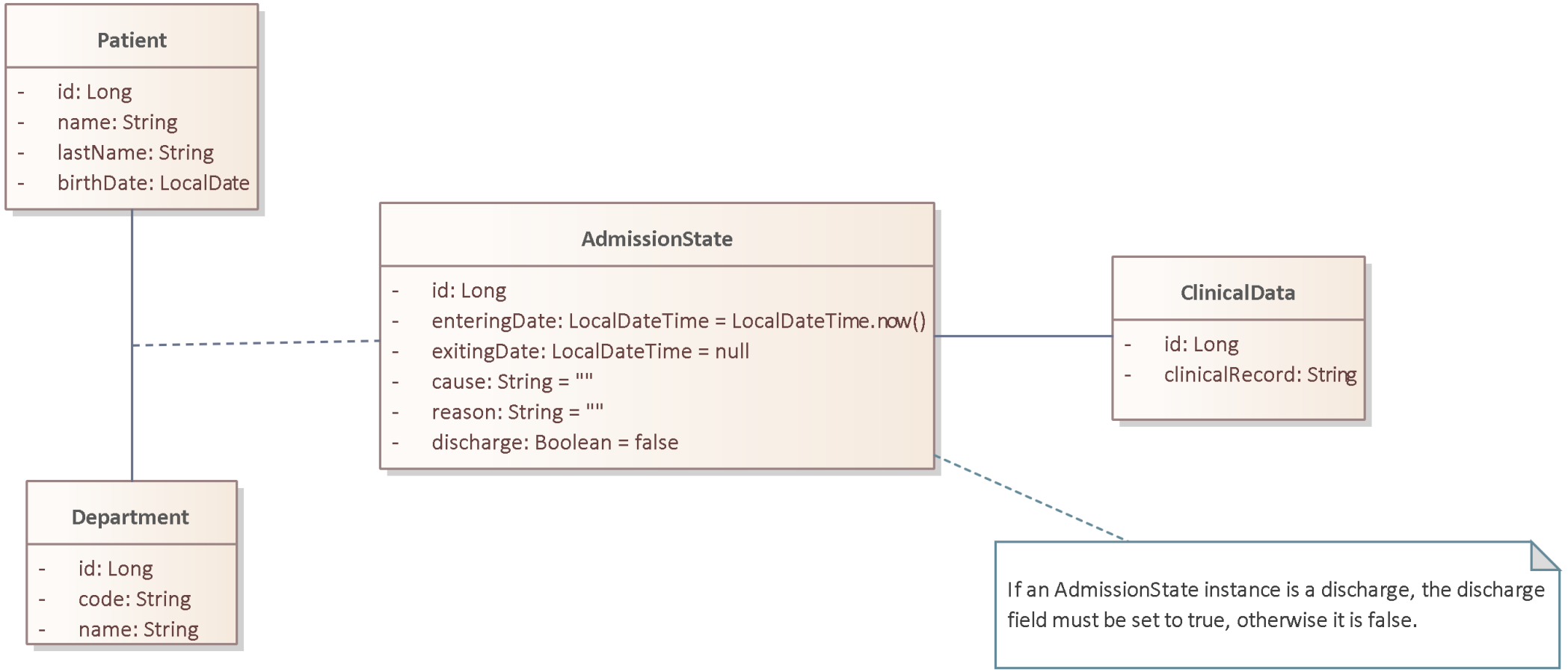


Manage Patient's Clinical Record



Model

Database Model



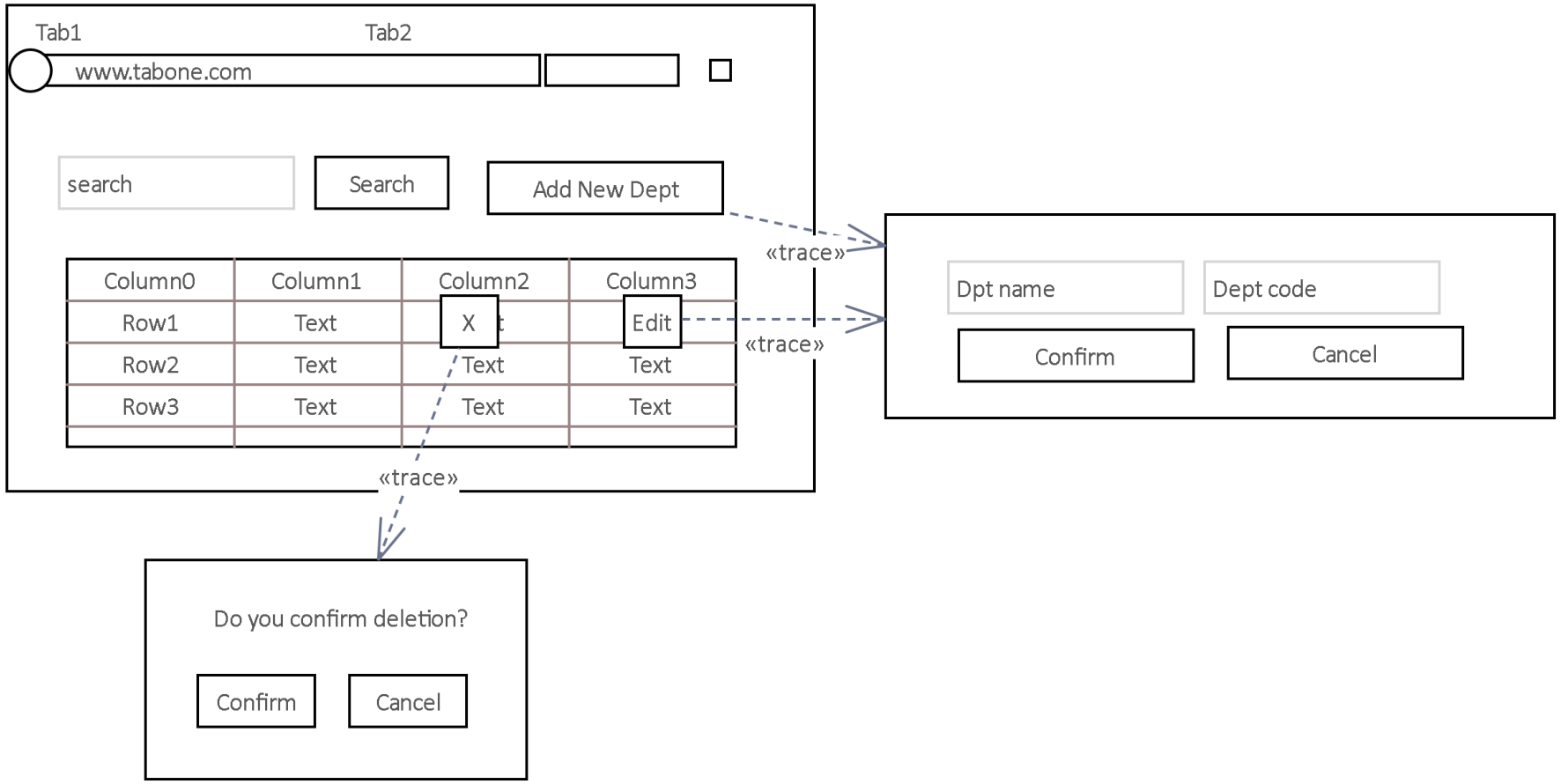
User Interfaces

User Interfaces Notes

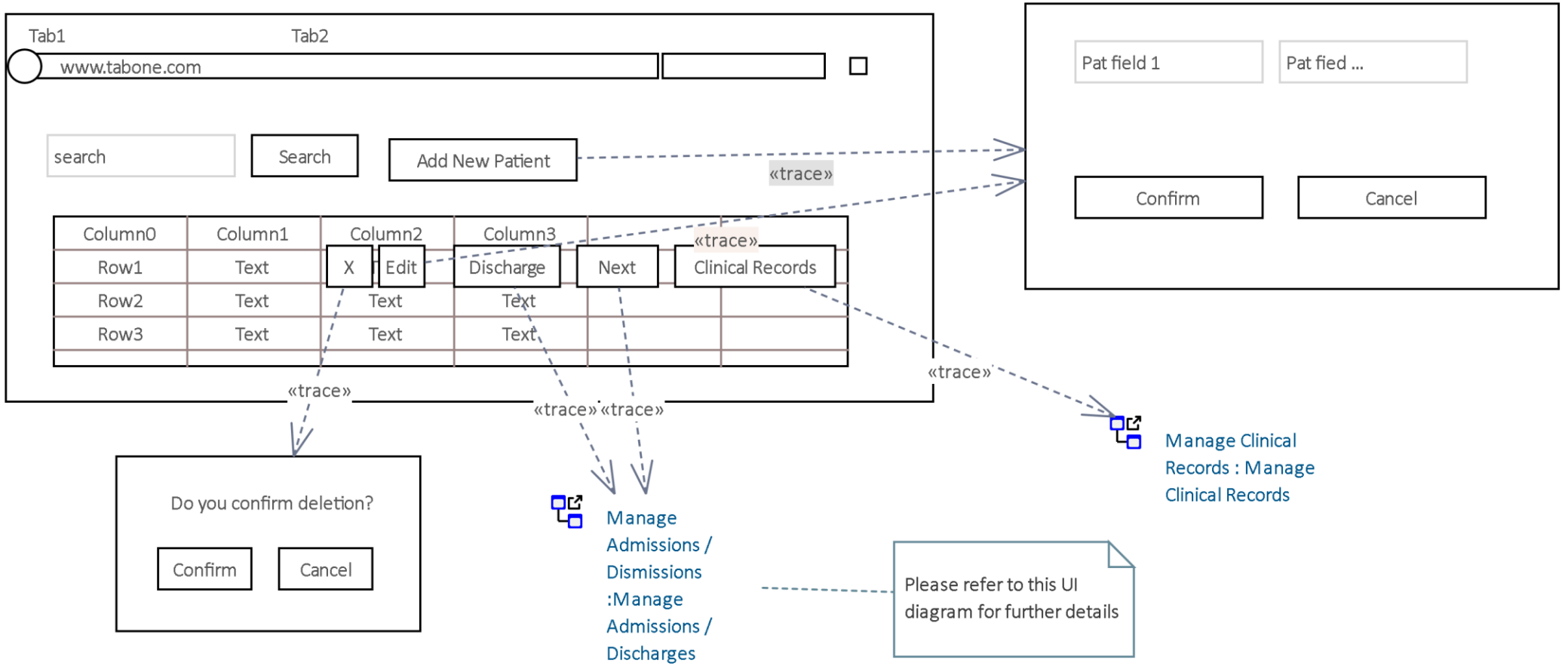
- The UIs depicted in the next slides are only examples. The group can rearrange the UIs if the following conditions are met:
  - The resulting behaviour is equal or better than without rearrangement
  - The new UI is responsive
  - All the use cases are implemented
- All the UIs must be responsive for at least three sizes of view area: small, medium, large (i.e. smartphone, tablet, monitor)

UI MUST NOT BE DEVELOPED IN THIS COURSE

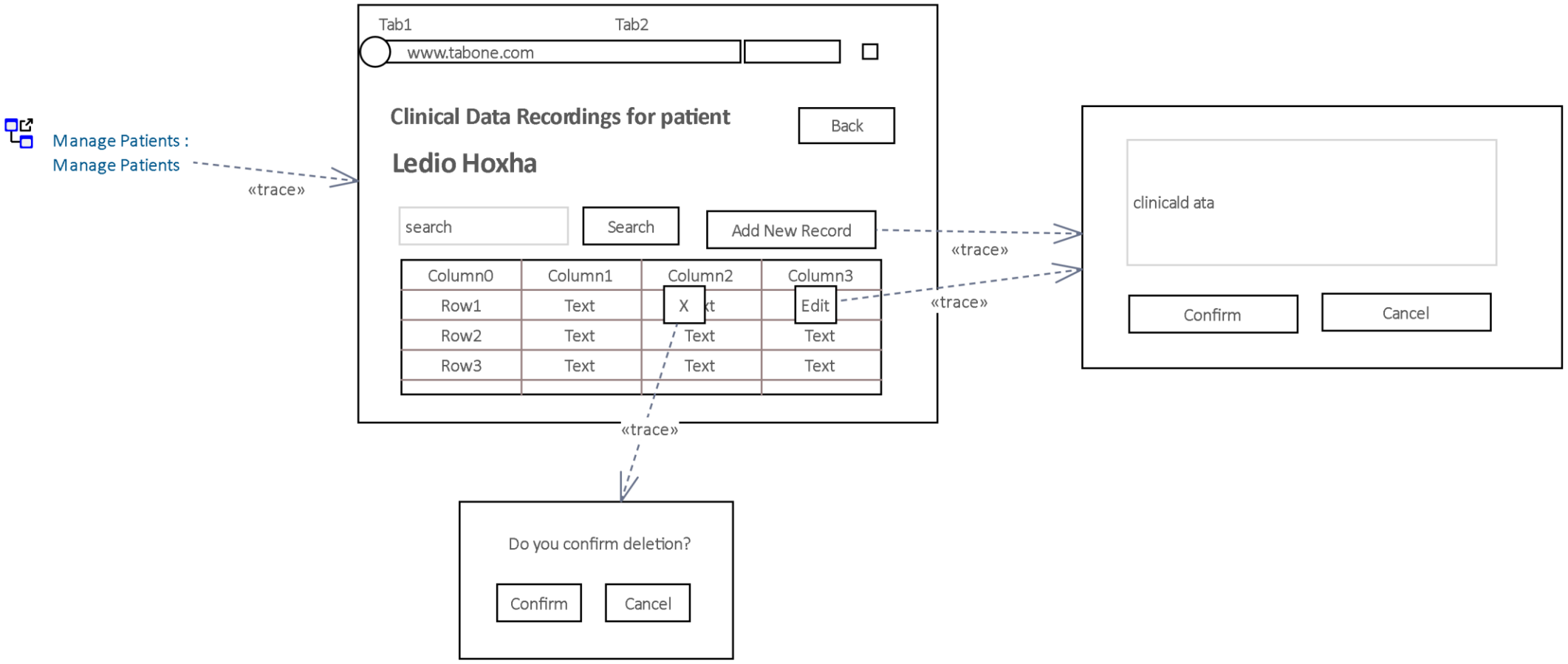
Manage Departments



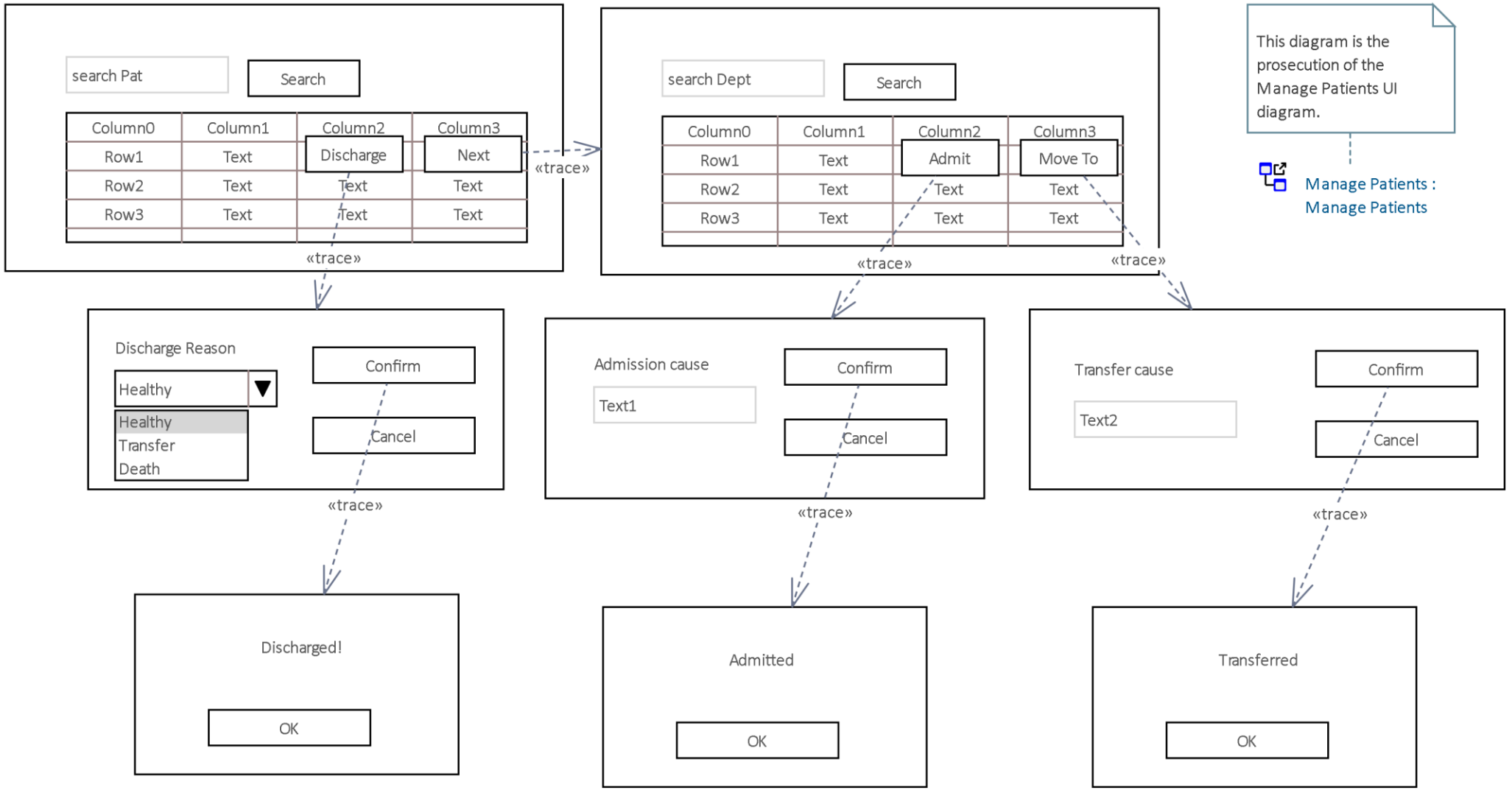
Manage Patients



Manage Clinical Records



Manage Admissions / Discharges

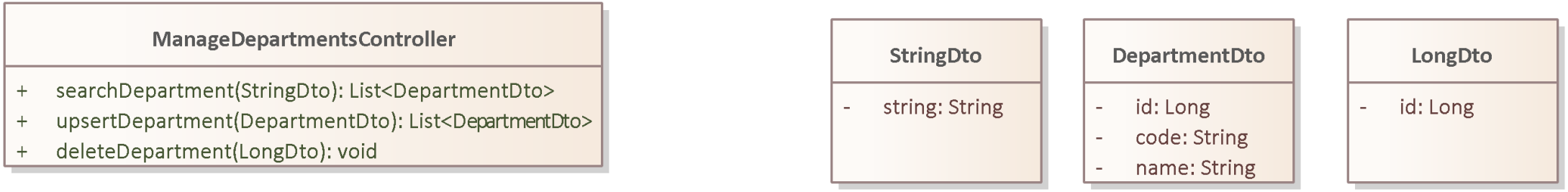


RESTful Services

Specifications Notes

- The following specification about the RESTful services to be implemented:
  - Are not necessarily all the needed services: some could be missing and must be defined by the students
  - Are related to the controller layer only and do not describe other layers, that must be fully designed by the students
  - Do not have any kind of security or authentication (that are not needed for the project work)
  - Can be modified by the students if they need different parameters or other changes (for example, the error management is completely missing …)
  - DTO specifications have been mostly omitted and DTOs must be defined by the students

Departments Management



Questions?

Thank you for your kind attention